Reliable Transaction Router

Migration Guide

Order Number: AA–R88LB–TE

June 1999

This guide explains how to migrate from Reliable Transaction Router™ (RTR) Version 2 to RTR Version 3 on OpenVMS™ systems, and provides information on new and obsolete features.

Revision/Update Information:	This guide supersedes the Reliable Transaction Router *Migration Guide* for Version 3.1D.
Operating System:	OpenVMS Versions 6.2, 7.1, 7.2
Software Version:	Reliable Transaction Router Version 3.2

Compaq Computer Corporation
Houston, Texas

First Printing, December 1997
Revised, June 1999

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

Preface

This guide explains how to migrate a Reliable Transaction Router (RTR) environment and RTR applications from RTR Version 2 to RTR Version 3. It assumes that the application continues to use the RTR Version 2 application programming interface (API) without change. It also provides information on new and obsolete features.

Audience

This guide is written for RTR system managers.

The system manager should be familiar with the following aspects of the OpenVMS operating system:

- DIGITAL Command Language (DCL)
- A text editor, such as EDT or EVE

The system manager should also be familiar with:

- Monitoring RTR performance
- Adjusting RTR performance
- Installation of RTR Version 2
- Making changes to the RTR environment
- Application use of the RTR Version 2 Application Programming Interface (API)
- Applications running on RTR

Organization of This Guide

The following list can help you find information in this guide:

Chapter 1	Provides an introduction to the migration guide and summarizes new and changed features.
Chapter 2	Describes changes to the installation procedure.
Chapter 3	Describes how the RTR architecture has changed.
Chapter 4	Describes changes to the network configuration that supports RTR.
Chapter 5	Describes changes important to the system manager.
Chapter 6	Describes changes to the RTR API.
Chapter 7	Describes performance and application tips.
Chapter 8	Describes problem diagnosis and reporting.

Related Documents

The following documents provide more information about topics discussed in this guide:

Document Title	Description
Reliable Transaction Router *Installation Guide*	Describes how to install Reliable Transaction Router.
Reliable Transaction Router *System Manager's Manual*	Describes how to configure, manage, and monitor Reliable Transaction Router
Reliable Transaction Router *Application Programmer's Reference Manual*	Describes how to code RTR applications, and contains full descriptions of RTR API calls.
Reliable Transaction Router *Application Design Guide*	Describes how to design applications that use RTR.
Reliable Transaction Router *Release Notes*	Describes new features, changes, and known restrictions for Reliable Transaction Router on all supported platforms.

The following document is your best source for information on OpenVMS procedures that you should be familiar with when using this migration guide:

Document Title	Description
OpenVMS System Manager's Manual: Essentials	Part one of a task-oriented guide to managing an OpenVMS system.

The following document is a useful source for writing applications on OpenVMS:

Document Title	Description
DEC C Run-Time Library Reference Manual for OpenVMS	Describes use of the DEC C run-time library.

Reader's Comments

Compaq welcomes your comments on this guide. Please send us your comments by email to *rtrdoc@compaq.com*. Include the title of the manual, date on the title page, section and page numbers with your comments or suggestions.

Conventions

The following conventions are used in this document.

Convention	Meaning
Ctrl/X	While you hold down the Ctrl key, press another key or a pointing device button.
Italic	Indicates parameters whose values you must provide. For example, if the procedure asks you to type *file name*, you must type the actual name of a file. Italic also indicates variables and the titles of referenced documents.

Convention	Meaning
`monospace`	Indicates the actual commands, words, or characters that you type in a dialog box, at a command prompt, or system output.
Note:	Provides information of special importance.
/	A forward slash in command descriptions indicates that a command qualifier follows.
. . .	A horizontal ellipsis following an entry in a command line indicates that the entry or a similar entry can be repeated any number of times. An ellipsis following a file name indicates that additional parameters, values, or information can be entered.
. . .	A vertical ellipsis in an example indicates that not all the data are shown.

1 Introduction

This document is intended to assist RTR Version 2 users to migrate to RTR Version 3.

1.1 Why Migrate?

Migration to RTR Version 3 takes advantage of the new features and improved capabilities of RTR Version 3. These include:

- Improved installation procedure using Polycenter Software Installation Utility (PCSI), not VMSINSTAL
- Interoperability with multiple operating systems including:
  - OpenVMS (Alpha and VAX)
  - UNIX (DIGITAL UNIX, SUN Solaris, IBM AIX, HP–UX)
  - Windows NT
  - Windows 95 (client only)
- Improved ability to choose a primary server
- Use of more than one other node for a standby server
- Network transport selection (DECnet or TCP/IP)
- New SHOW command screens containing information about transactional messages, resource managers, and some new state names
- New MONITOR pictures
- A new API, the Portable API
- Role clarification (Requestor changed to Client)
- Compatibility with Version 2 applications (no recompilation or linking required)
- Use of mailboxes instead of OpenVMS cache and global sections for interprocess communication
- Use of OpenVMS quotas, eliminating need for manually sizing configuration parameters
- Enhanced partition management
- Enhanced journal management
- Exception logging
- Support for industry standard protocols for resource managers:
  - Microsoft DTC (Distributed Transaction Communicator) support
  - XA protocol

- Support for subtransactions or nested transactions

Additionally, other considerations are:

- New features will be implemented in RTR Version 3, not in RTR Version 2
- Some software problems will be addressed only in RTR Version 3 and not in RTR Version 2
- Some improved capabilities are already available only in RTR Version 3
- More extensive release test coverage than with RTR Version 2
- RTR Version 3 tested for year 2000 compliance (RTR Version 3.1D, Version 3.2)
- Any residual year 2000 problems will be fixed only in RTR Version 3.1D or later

Other changes introduced with RTR Version 3:

- New format and content of journal to improve security and flexibility of transactional messaging
- Change of shared library name RTRSHR.EXE to LIBRTR.EXE

Note

There is no upgrade path for Windows 3.1 clients; applications must be rewritten using the RTR Version 3 API.

1.2 Goals and Nongoals

The goal of this document is to assist the RTR Version 2 system manager in planning and implementing the migration of an RTR Version 2 environment to RTR Version 3.

It is not a goal of this document to instruct the system manager about RTR or teach troubleshooting or analysis of the RTR environment.

1.3 Reading Guidelines

Read this document, the RTR Version 3 documentation and Release Notes before beginning to implement an RTR migration to RTR Version 3.

2 Installation

The installation for RTR Version 3 has changed significantly from Version 2. In Version 2, the installation tool was VMSINSTAL; for Version 3, the installation tool is PCSI. Follow instructions in the Reliable Transaction Router *Installation Guide* to perform your RTR Version 3 installation.

Note

Reading Release Notes in RTR Version 3 is different from in RTR Version 2. See the Reliable Transaction Router *Installation Guide* for information on installing the product and reading release notes.

In a cluster environment, a planned transition from RTR Version 2 to Version 3 could be done as follows:

1. Install RTR Version 3 on a single standalone node.
2. Bring up RTR Version 3 on the standalone node with the RTR application.
3. Verify that the application and RTR Version 3 work as expected. Resolve any problems before proceeding.
4. Stop all transactions and RTR with the following commands:

   ```
   $ RTR STOP RTR
   $ RTR DISCONNECT SERVER
   $ RTR DUMP JOURNAL
   ```

5. Examine the output of the DUMP JOURNAL command to verify that all transactions have been flushed from the journal.
6. Bring down RTR Version 2 on the entire cluster.
7. Install RTR Version 3 on the cluster.
8. Start up RTR Version 3 on each node.
9. Verify that the application is running correctly on each node.
10. Verify that the application is running correctly on the cluster.
11. Verify that the application is running correctly throughout the RTR facility and network environment.

2.1 Cleaning Up the Old Version 2 Environment

Before bringing up the RTR Version 3 environment, you will need to shut down RTR Version 2 on client systems, let the RTR journal file clear, and then bring up RTR Version 3. This should be part of the process you use in your planned migration. See Section 2.6 for more information on checking journal files.

2.2 Preserving the Old Environment

Applications that run in the RTR Version 2 environment on OpenVMS systems will run in the RTR Version 3 environment on OpenVMS systems. However, as part of your testing and verification of the new environment, you should check that an RTR Version 2 application runs as expected after the upgrade.

You cannot mix RTR Version 2 and Version 3 routers and back-end systems; all routers and back-end systems in a facility must be either RTR Version 2 or RTR Version 3. Front-end systems can be either Version 2 or Version 3.

2.3 Can Both RTR Version 2 and Version 3 Coexist On the Same Node?

No, RTR Version 2 and Version 3 cannot coexist on the same node.

2.4 Can RTR Version 2 and Version 3 Applications Coexist on the Same Node?

Under most circumstances, RTR Version 2 applications will run in the RTR Version 3 environment unchanged, and RTR Version 2 and Version 3 applications can coexist on the same node, and exchange messages.

Because the RTR Version 2 API is frozen, any new features requiring a change in the API cannot be exploited from within an RTR Version 2 application. This may be a reason to consider migration. Additionally, if portability is an issue, migrate to RTR Version 3.

2.5 Process Quotas

RTRACP buffers all active transactions. All message information that was previously kept in the shared memory RTR cache is now kept within RTRACP process memory.

Because of the use of mailboxes to handle message traffic, mailbox quotas should generally be set larger than in Version 2. These quotas or limits include:

- AST queue limit
- Byte count limit for both the application and ACP
- Buffered I/O count limit
- Direct I/O count limit
- Paging file limit
- Timer queue entry limit

Table 2–1 provides estimates of values for these limits.

Note

Values in Table 2–1 supersede values previously documented.

Table 2–1 OpenVMS Limits

Limit Name	OpenVMS Name	Value for ACP	Value for Application
AST queue limit	ASTlm	2000 or more	
Byte count limit	Bytlm	32K + (32K * *appn* †)	Not less than 100K
Buffered I/O count limit	BIOlm	Not less than 3 * *appn*	
Direct I/O count limit	DIOlm	2000 or more	
Paging file limit	Pgflquo	Not less than 200K	
Timer Queue Entry limit	TQElm	2000 or more	

†In the table, *appn* is the number of application processes.

For more information on these limits, see the *OpenVMS System Manager's Manual: Essentials*.

2.6 Journal Issues

With RTR Version 3 software, the format and content of the transaction journal have changed. In general, if you upgrade or migrate to RTR Version 3 but continue to use the RTR Version 2 API and DECnet, you can use your existing application without change, However, if an RTR Version 2 application stored and used a transaction ID, the changed transaction ID format of RTR Version 3 can cause problems to that application. To correct such a problem, change the application.

A journal file resides on each RTR back-end system. This is not a change from RTR Version 2. Before you shut down RTR Version 2 to bring up RTR Version 3, you must deal with your journal file, using the following procedure:

1. In the Version 2 environment, stop all clients so that no new transactions can be initiated.
2. Monitor the Version 2 journal file to check the status of all current transactions.
3. When all transactions are complete and the journal file is empty, you can delete the journal file at this point, and shut down the entire Version 2 environment as follows:
   a. Shut down RTR applications.
   b. Shut down RTR Version 2.
   c. Install RTR Version 3.
   d. Bring up RTR Version 3 including performing the following tasks:
      i Start RTR Version 3.
      ii Create the new journal file.
      iii Create the new operating environment, for example, define facilities.
      iv Start the application.

2.6.1 Removing the Old Journal

To verify that the new journal is running correctly, use the DUMP JOURNAL command to verify that transactions are processing as expected, and to be sure that all transactions have completed before bringing down RTR to install RTR Version 3.

There is no need to save the old journal file once all transactions have cleared.

2.6.2 Journal Compatibility

RTR Version 2 journal files are incompatible with RTR Version 3 journal files. No tools exist to migrate RTR Version 2 journal files to the RTR Version 3 journal file.

2.6.3 Location and Naming Conventions

With RTR, the journal records transactions for use during recovery when required. You specify the disk location for the journal file with the CREATE JOURNAL command. This is unchanged for RTR Version 3.

However, the location and naming conventions for the journal have changed for RTR Version 3.

In RTR Version 2, journal files reside on each back-end node in:

dev:[RTRJNL]*filename*

In RTR Version 3, journal files reside on each back-end node in:

dev:[RTRJNL.SYSTEM]*nodename*.J*nn*

Group names are used for naming RTR journal files.

2.7 Rights and Privileges

You need the same rights and privileges to manage the RTR environment and RTR applications in Version 3 as in Version 2.

To manage RTR, you must have one of the following OpenVMS system rights or privileges: OPER, SETPRV, RTR$OPERATOR. To use the RTR API rtr_request_info, you must have the following right: RTR$INFO.

To run an application, you must have the following OpenVMS privilege: TMPMBX.

2.8 Memory and Disk Requirements

Generally, RTR Version 3 may make more demands on system memory than RTR Version 2, which can cause performance reduction. Adding more memory may be useful in improving performance.

Table 2–2 lists the OpenVMS requirements for space on the system disk. These sizes are approximate; actual size may vary depending on system environment, configuration, and software options. For additional details, see the Reliable Transaction Router for OpenVMS Software Product Description.

Table 2–2 OpenVMS Disk Requirements

Requirement	RTR Version 2	RTR Version 3
Disk space (installation)	40,000 blocks (20MB)	50,000 blocks (25MB)
Disk space (permanent)	24,000 blocks (12MB)	36,000 blocks (18MB)

2.9 Rollback to RTR Version 2

To restore the RTR Version 2 environment if RTR Version 3 does not work with your applications as expected, use the following procedure:

1. In the RTR Version 3 environment, stop all clients so that no new transactions can be initiated.
2. Monitor the RTR Version 3 journal to check the status of all current transactions. Once all transactions are complete and the journal is empty, you can proceed to the next step.
3. When all transactions are complete, and the journal file is empty, delete the journal, and shut down the entire RTR Version 3 environment as follows:
   a. Shut down RTR applications.
   b. Shut down RTR Version 3.

   ```
   $ RTR STOP RTR
   $ RTR DISCONNECT SERVER
   ```

   c. Remove RTR Version 3 by issuing the following command:
   `$ product remove RTR`
   d. Deassign the logical name RTRSHR with the OpenVMS DCL command:
   `DEASSIGN/SYSTEM/EXEC "RTRSHR"`
   e. Install RTR Version 2 using VMSINSTAL.
      – Answer yes to the VMSINSTAL questions on purging files replaced by the installation, and running the IVP.
      – If you receive the message `"The IVP has completed successfully"`, RTR has been successfully installed.
   f. Bring up RTR Version 2 including performing the following tasks:
      i Start RTR Version 2.
      ii Create the RTR Version 2 journal.
      iii Create the RTR Version 2 operating environment.
      iv Start the RTR Version 2 application.

3 Architectural Changes

RTR Version 3 introduces certain process and other architectural changes. The following sections highlight these changes.

3.1 RTR Daemon Process

In RTR Version 3, a new RTR daemon process (called RTRD) is used by the RTRACP process to build TCP/IP connections for internode links. The RTR daemon process is present only on systems with IP networking installed and with IP enabled as an RTR transport (see Chapter 4, Network Issues, for information on setting your network transport).

3.2 Command Server Process

The command server process name is RTRCSV_<*username*>.

In RTR Version 2, a command server was started for every user login invocation to RTR to enter operator commands. With RTR Version 3 there is one command server per node for each user logged in through a common user name.

Note

Command server timeouts are the same in RTR V3 as in RTR V2.

3.3 The Shared Library (LIBRTR.EXE)

In RTR Version 3, LIBRTR supersedes RTRSHR. This library module LIBRTR contains most of the RTR code, in contrast with RTR Version 2, where only RTR code specific to application context was contained in RTRSHR. All RTR Version 2 binaries have been superseded by the two executables LIBRTR.EXE and RTR.EXE, in RTR Version 3. Table 3–1 shows the executables of RTR Version 2 and Version 3.

Table 3–1 RTR Executables

RTR Version 2	RTR Version 3
RTRSHR	LIBRTR
RTR	RTR
RTRCOMSERV	Now part of LIBRTR.
RTRACP	Now part of LIBRTR.

(continued on next page)

Table 3–1 (Cont.) RTR Executables

RTR Version 2	RTR Version 3
RTRRTL	No longer apply.

3.4 The ACP Process

The RTR Application Control Process (ACP) handles application control, and has the process name RTRACP. This is unchanged from RTR Version 2.

3.5 Interprocess Communication

In RTR Version 2, global sections (cache) were used for interprocess communication. In RTR Version 3, interprocess communication is handled with mailboxes. Each RTR process, including any application process, has three mailboxes to communicate with the RTRACP process:

- Read
- Write
- Beacon/Control

3.6 Shared Memory Parameters

With RTR Version 2, the SHOW RTR/PARAMS command showed the following:

- Maximum links
- Maximum facilities
- Maximum relations
- Maximum processes
- Cache pages

The /PARAMS qualifier is obsolete in RTR Version 3, and the parameters it showed no longer apply. In RTR Version 3, these parameters are handled with OpenVMS mailboxes, which you can check using OpenVMS procedures. See the *OpenVMS System Manager's Manual: Essentials* for more information.

3.7 Counters

In RTR Version 2, shared memory in global sections was directly accessible using the RTR command server. In RTR Version 3, process counters are still kept in shared memory, but they are accessed from the command server through RTRACP. Thus, accessing these and other counters involves communicating with RTRACP. Other counters are contained within the address space of the ACP.

3.8 Quorum Issues

Network partitioning in RTR Version 3 is based on a router and backend count, whereas in RTR Version 2 it was based on quorum. However, quorum is still used in RTR Version 3; state names and some quorum-related displays have changed.

Additionally, the quorum-related condition of a node in a minority network partition is handled more gracefully in RTR Version 3. In RTR Version 2, a shadowed node in a minority network partition would just lose quorum; in RTR Version 3, the MONITOR QUORUM command states that the node is "in minority," providing more information. The algorithms used to determine quorum have also changed significantly to allow a more stable traffic pattern.

3.9 Server-Process Partition States

As in RTR Version 2, there are three server-process partition states:

- Primary
- Standby
- Shadow

With RTR Version 3, a server process that is initially the primary in a standby or shadow environment returns to primary after recovery from a network loss. (With RTR Version 2, there was no way to specify which node would become the primary after network recovery.) Unlike RTR Version 2, where the location of a primary server after a network outage was unpredictable, as long as servers have not been restarted and both servers are accessible, RTR Version 3 retains the original roles.

With RTR Version 3.2, RTR provides commands such as SET PARTITION /PRIMARY that the operator can use to specify a process partition state.

4 Network Issues

With RTR Version 3, two network transports are available:

- DECnet (default on OpenVMS)
- TCP/IP

At least one transport is required. If a destination supports both transports, RTR Version 3 can use either.

Any node can run either protocol, but the appropriate transport software must be running on that node. For example, for a node to use the DECnet protocol, the node must be running DECnet software. (For specific software network version numbers, see the RTR Version 3 OpenVMS Software Product Description.)

A link can fail over to either transport within RTR. Sufficient redundancy in the RTR configuration provides greater flexibility to change transports for a given link when necessary.

4.1 DECnet Support

With RTR Version 2, the only transport was DECnet Phase IV; it provided DECnet Phase V support but without longnames. With RTR Version 3, both DECnet Phase IV and DECnet-Plus (DECnet/OSI or DECnet Phase V) are supported, including support for longnames and long addresses.

4.2 TCP/IP Support

DECnet-Plus and TCP/IP provide multihoming capability: a multihomed IP node can have more than one IP address. RTR does name lookups and name address translations, as appropriate, using a name server. To use multihomed and TCP/IP addresses, Compaq recommends that you have a local name server that provides the names and addresses for all RTR nodes. The local name server should be available and responsive.

Name servers for all nodes used by RTR should contain the node names and addresses of all RTR nodes. Local RTR name databases must be consistent.

Note

Include all possible addresses of nodes used by RTR, even those addresses not actually used by RTR. For example, a node may have two addresses, but RTR uses only one. Include both addresses in the local name database.

4.3 Specifying a Preferred Transport

During installation, the system manager can specify either transport, using logical names `RTR_DNA_FIRST` or `RTR_TCP_FIRST`. For example, in the RTR$STARTUP.COM file (found in SYS$STARTUP), the following line specifies DECnet as the default transport:

```
 $ DEFINE/SYSTEM RTR_PREF_PROT RTR_DNA_FIRST
```

To set the default transport to TCP/IP, remove (comment out) this definition from RTR$STARTUP.COM and restart RTR. For the change to take immediate effect, you must undefine the old logical name before restarting RTR.

You can also change the above command in RTR$STARTUP.COM to the following:

```
$ DEFINE/SYSTEM RTR_PREF_PROT RTR_TCP_FIRST
```

When creating a facility using TCP/IP as the default, you can specify dna.*nodename* to override TCP/IP and use DECnet for a specific link. Similarly, when using DECnet as the default, you can specify tcp.*nodename* to use TCP/IP for a specific link. If the wrong transport has been assigned to a link, you must trim all facilities to remove nodes using the link (use the TRIM FACILITY command) to remove the link, then add the nodes back into the facility specifying the correct transport.

To run the DECnet protocol exclusively, use the following definition for the RTR preferred protocol logical name:

```
$ DEFINE/SYSTEM RTR_PREF_PROT RTR_DNA_ONLY
```

For examples of this command syntax, see the section on Network Transports in the Reliable Transaction Router *System Manager's Manual*.

4.3.1 Supported Products

Network products supported are listed in the RTR Version 3 Software Product Description.

5
System Management

A number of changes that affect system management have been introduced with RTR Version 3. The following sections describe these changes.

5.1 OpenVMS Quotas

RTR Version 2 used OpenVMS quota values specified on the RTR START command or calculated defaults. Because RTR Version 3 uses dynamic allocation (with the exception of the number of partitions that is statically defined), RTR does not calculate the required quotas, but depends on the system manager to configure quotas adequately. The value of maximum partitions is now set at 500. (See the RTR *Release Notes* for further information on partitions.)

For example, with RTR Version 2 you were required to explicitly specify the number of links or the number of facilities if defaults were too low. You no longer need to specify each RTR parameter value manually. Additionally, because RTR Version 3 uses mailboxes, you use the appropriate OpenVMS quotas to establish sufficient resources to support RTR Version 3 interprocess communication.

In RTR Version 3, all these parameters are governed by OpenVMS quotas. To establish these for RTR Version 3, DIGITAL recommends that you record the actual quotas used by RTR Version 2 on each node and add 50 percent to these values for RTR Version 3. See Table 2–1 for some specifics.

5.2 Startup

There is a new RTR$STARTUP.COM file in SYS$STARTUP. It contains several changes including specifying RTR file locations, and choice of transport (protocol).

5.3 Creating Facilities

You create facilities the same way in RTR Version 3 as in RTR Version 2

5.3.1 Naming Nodes

With the addition of TCP/IP and DECnet-Plus (DECnet/OSI) to RTR Version 3, you can now use longnames for node names.

5.3.2 Modifying Facility Configurations

To modify facilities, you use the same procedures in RTR Version 3 as in RTR Version 2. One facility command has been changed:

```
SET FACILITY/BROADCAST=MINIMUM=n
```

has been changed to:

```
SET FACILITY/BROADCAST_MINIMUM_RATE=n
```

5.4 Interoperability

All supported operating systems can interoperate together in the RTR environment, as described in Table 5–1.

Table 5–1 Interoperability Between Nodes

RTR Version 3 nodes interoperate with...	Description
Other RTR Version 2 nodes	In RTR Version 3, RTR uses data marshalling (examination of byte format of messages) and can handle data of more than one byte format, making the appropriate translation as required. However, an application running with RTR may not adequately handle different the byte formats used on different hardware architectures. RTR Version 3 lets you run both RTR Version 2 and RTR Version 3 nodes in the same environment, but because the RTR Version 2 API does not have the data marshalling capability, an RTR Version 2 application must deal with the different data formats.
Other RTR Version 3 nodes	RTR Version 3 is fully compatible with other nodes running RTR Version 3. See the RTR Version 3 *Release Notes* for specifics on known requirements and restrictions.

5.5 Monitoring

Several screens that provide dynamic information on transactions and system state have changed for RTR Version 3, as described in the following sections.

5.5.1 RTR Version 2 Screens

Table 5–2 lists the RTR Version 2 screens that are no longer available in RTR Version 3. In general, information in these monitor pictures is no longer applicable. For example, there is no longer a need to examine cache, because RTR Version 3 deals with memory management using OpenVMS mailboxes.

Table 5–2 Obsolete RTR Version 2 Monitor Pictures

bequorum	cache	chmdata
chmmsg	declare	delayproc
dtinfo	failure	memory
msgacpsys	packets	process
toptps	trquorum	

5.5.2 New Screens

Table 5–3 lists the monitor screens that are new to RTR Version 3.

Table 5–3 New RTR Version 3 Monitor Pictures

Picture	Description
accfail	Shows most recent links on which a connection attempt was declined.
acp2app	Shows RTRACP-to-application message counts.
app2acp	Shows application-to-RTRACP message counts.
broadcast	Shows information about RTR user events by process.
connect	Renamed to netstat.
connects	Shows connection status summary.
dtx	Shows distributed transaction calls status.
dtxrec	Shows distributed transaction recovery status and journal states.
event	Shows event routing data by facility.
frontend	Shows frontend status and counts by node and facility, including frontend state, current router, reject status, retry count, and quorum rejects.
group	Shows server and transaction concurrency by partition.
ipc	Shows interprocess communication counts for messages and connects.
jcalls	Displays counts of successful (success), failed (fail), and total journal calls for local and remote journals.
netstat	Shows link counters relating to connection management, with link state and architecture of remote nodes.
rdm	Shows memory used by each RTR subsystem.
rejects	Displays the last `rtr_mt_rejected` message received by each running process.
rejhist	Displays the last ten `rtr_mt_rejected` messages received by each running process.
response	Displays the elapsed time that a transaction has been active on the opened channels of a process.
rfb	Displays router failback operations, both a summary and detail by facility.
routing	Displays statistics of transaction and broadcast traffic by facility.
rscbe	Displays the most recent calls history for the RSC subsystem on a backend node.
system	Displays high level summary of critical monitor pictures.
tpslo	Shows transaction commits by process.
trans	Displays transactions by frontend, facility, and user for each frontend, router, and backend.
V2calls	Shows RTR Version 2 system service calls.
XA[1]	Displays status of XA interface activities.

[1]UNIX and NT only

5.5.3 User Parsing of Monitor Output

Because of changes to many monitor screens with RTR Version 3, user scripts that parse monitor output may need to be reviewed and changed.

5.5.4 User Customized Monitors

User monitors customized for use with RTR Version 2 may or may not work with RTR Version 3. DIGITAL recommends that user-customized monitors be tested with RTR Version 3 before being put into production.

5.5.5 History Screens

New monitor screens that show partition state or network connection history include MONITOR accfail and MONITOR rscbe.

5.6 Remote Command Support

With the new support for TCP/IP, you can execute commands on remote systems using the rsh utility.

To use this feature, check your operating system documentation for how to ensure access to a TCP/IP environment, and grant such privileges to users. You may, for example, need to make an entry in the .rhosts file in the home directory of the RTR user on the target node or nodes, among other things. This file would contain the host name (and optionally the user name) of the node where the remote commands will be issued.

5.7 Partition Management

Partitions are now managed objects in RTR Version 3.2; partitions can be defined by the system manager before application startup. New commands listed in Table 5–6 support partition management.

5.8 Transaction State Management

Transaction state can be changed by the system manager to resolve certain deadlock situations using the SET TRANSACTION command. For more information on this command, see the RTR *System Manager's Manual*.

5.9 Using RTR Version 2 Command Procedures

Most RTR Version 2 command procedures will still work with RTR Version 3. One changed command is listed in Section 5.3.2 and fully described in the RTR *System Manager's Manual*.

5.10 Command Line Interface Support for RTR Version 2 API

Command-line interface support for the RTR Version 2 API may not be fully compatible between RTR Version 2 and RTR Version 3. See the RTR Version 3 *System Manager's Manual* and *Release Notes* for additional details.

5.11 Interpreting Output from SHOW Commands

The output from SHOW commands has changed with RTR Version 3. All SHOW output now includes date and time. Additional changes are listed in Table 5–4.

Table 5–4 Changed SHOW COMMANDS

Command	Description
SHOW CLIENT	This new command displays information about client channels. It supersedes the SHOW REQUESTOR command.
SHOW SERVER/FULL	The SHOW SERVER/FULL command provides new information on states.

(continued on next page)

Table 5–4 (Cont.) Changed SHOW COMMANDS

Command	Description
SHOW TRANSACTION	With the SHOW TRANSACTION command, you can now specify display for a frontend, backend, or router.
SHOW FACILITY/LINK	The SHOW FACILITY/LINK command provides new information on states.
SHOW PARTITION /FULL	The SHOW PARTITION/FULL command provides new information on states.

5.12 Comparing RTR Version 2 and Version 3 Utility Commands

Table 5–5 lists obsolete RTR Version 2 commands; they do not appear in RTR Version 3. In general, they no longer apply.

Note

In a mixed RTR Version 2 and Versio n3 environment, you cannot execute commands remotely from a Version 3 to a Version 2 system, or vice versa, with the /NODE qualifier.

Table 5–5 Obsolete OpenVMS RTR Utility Commands

Command Name	Comparable RTR Version 3 Command
ATTACH	No equivalent.
DEFINE/KEY	No equivalent.
PRINT	Use MONITOR *filespec*/OUTPUT=*filename*, and the OpenVMS PRINT command.
SHOW RTR/PARAMS	No equivalent.

Table 5–6 lists commands that are new to RTR Version 3. These commands are described more fully in the Reliable Transaction Router *System Manager's Manual*.

Table 5–6 New OpenVMS RTR Utility Commands

Command Name	Description
CALL RTR_<*routine*>	Executes the named routine and returns status.
CREATE PARTITION	Creates a named partition.
DELETE PARTITION	Deletes a named partition.
DISPLAY STRING	Displays a string in a monitor picture.
DUMP JOURNAL	Displays contents of RTR journal.
EXECUTE	Executes RTR commands from a file.

(continued on next page)

Table 5–6 (Cont.) New OpenVMS RTR Utility Commands

Command Name	Description
QUIT	Exits RTR.
REGISTER RM[1]	Registers resource managers with the current transaction manager.
SET PARTITION	Sets partition properties.
SET TRANSACTION	Changes the state of a transaction.
SHOW CLIENT	Supersedes SHOW REQUESTOR.
SHOW RM[1]	Displays information for registered resource managers.
UNREGISTER RM[1]	Deletes a resource manager.

[1]UNIX and NT only

6 Running Version 2 Applications

In this chapter the term OpenVMS API refers to the Reliable Transaction Router for OpenVMS Version 2. The term Portable API refers to the API used in Reliable Transaction Router for OpenVMS Version 3.

With RTR Version 3, the Portable API provides:

- Portability over a wide range of language and machine environments
- Simplified handling of concurrency, independent of the type of operating system
- Support for communication between machines with different hardware representations of common data types
- Interoperability with existing applications that use the OpenVMS API
- Improved performance for commonly used transaction types
- Support for use within a threaded environment
- Support for foreign protocols such as XA and Microsoft DTC, with nested or subtransaction transaction capability

6.1 Comparison of OpenVMS API and Portable API

Table 6–1 lists the OpenVMS API and comparable Portable API calls.

Table 6–1 OpenVMS API and Portable API Comparison

OpenVMS API (Version 2)	Portable API (Version 3)
$dcl_tx_prc( )	rtr_open_channel( )
$start_tx( )	rtr_start_tx( ) [optional]
$commit_tx( )	rtr_accept_tx( )
$abort_tx( )	rtr_reject_tx( )

$vote_tx( )	rtr_accept_tx( )
	rtr_reject_tx( )

$deq_tx( )	rtr_receive_message( )

$enq_tx( )	rtr_send_to_server( )
	rtr_reply_to_client( )
	rtr_broadcast_event( )

$dcl_tx_prc( ) (SHUT)	rtr_close_channel( )
$get_txi( )	rtr_request_info( )
$set_txi( )	rtr_set_info( )
ASTPRM (on asynch calls)	rtr_set_user_handle( )
-	rtr_error_text( )
-	rtr_get_tid( )
-	rtr_prepare_tx( )
-	rtr_set_wakeup( )

OpenVMS API calls are obsolete and supported only on OpenVMS systems.

6.2 Recompiling and Relinking

There is no need to recompile and relink RTR Version 2 applications to run them on RTR Version 3.

To link RTR application programs, include the following line in the linker options file:

```
SYS$SHARE:LIBRTR/SHARE
```

An existing RTR Version 2 application will run on RTR Version 3.

However, if the application is recompiled, you must supply all parameters for any RTR call. For example, the $ENQ_TX service has eleven parameters, some of which were optional in RTR Version 2. All eleven must be supplied if the application is recompiled with RTR Version 3.

Note

If recompiling an RTR Version 2 application not written in C, use appropriate include files from the RTR Version 2 kit to ensure correct compilation. With the RTR Version 3 API, C is the only language for which a header file is provided.

6.2.1 RTR Version 2 Applications Running on RTR Version 3

- Linking Version 2 applications

  Existing RTR Version 2 applications will run if they have been linked against RTRSHR. (RTRSHR has been superseded by LIBRTR.EXE. Existing RTR Version 2 application executables will run without relinking since RTR$STARTUP.COM defines RTRSHR as a logical name that points to LIBRTR.EXE.)

  However, as RTRSHR.EXE is no longer distributed, change the linker options file referencing RTRSHR (that is, change SYS$SHARE:RTRSHR/SHARE to SYS$SHARE:LIBRTR/SHARE). After making this change, you can remove SYS$SHARE:RTRSHR.EXE from your system.

  If you are linking on a system where RTR Version 2 was never installed, always use SYS$SHARE:LIBRTR/SHARE.

- Event status

  With RTR Version 2, an application could pass event status as a parameter when calling $ENQ with the RTR$M_BROADCAST flag set. This broadcast $ENQ was delivered with the event status stored in the RTR$L_EVT_STATUS field of the RTR$_EVT data structure, and passed as a parameter to the broadcast AST.

  When running RTR Version 2 applications on RTR Version 3, event status is not passed from sender to receiver. All User events received by an RTR Version 2 application have the event status parameter set to 0 as shown in the following table:

If the sender is:	Then:
A Version 2 application	RTR Version 3 does not pass event status.
A Version 3 application	There is no event status.

- Channel number

  In some cases, RTR Version 2 returned the error status RTR$_INVALCH if an operation was attempted using an invalid channel number (for example, 0), and RTR$_CHNOTALLOC for potentially valid channel numbers that have not been declared. RTR Version 3 always returns RTR$_CHNOTALLOC.

- RTR STOP/ABORT

  If an RTR STOP RTR/ABORT command is issued with RTR Version 2, RTR executes an AST in the context of any applications that have an RTR channel open. The applications exit with the status RTR$_RTRWASSTO.

  In RTR Version 3, there is no difference in behaviour between the commands STOP RTR and STOP RTR/ABORT. If either of these commands is entered, RTR is always stopped. Any application with an RTR channel open receives an error status on any RTR operation in progress on that channel, but the application is not terminated. The application must handle the error correctly. (The error status returned in this case is RTR$_NOACP, the same status that is returned if the RTR ACP fails for any reason.)

6.3 Running Applications Installed with Privileges

With RTR Version 2, RTR calls execute in kernel mode; with RTR Version 3, RTR runs in application process mode, normally user mode.

6.3.1 Running Clients That Share Channels

With RTR Version 2, clients that start up and declare channels could use the flag INHNOSRVWT (inhibit-no server-wait) to proceed without waiting. (It lets $DCL_TX_PRC/REQ complete before servers have been declared.) With RTR Version 3, to perform a similar operation, an application must have either the OPER or RTR$OPERATOR process right.

6.4 Application Level Interoperability

With RTR Version 2, application level interoperability worked only with both applications running on the same operating system. With the upgrade to RTR Version 3, applications using the RTR Version 3 API can run on any supported operating system, and RTR Version 2 and RTR Version 3 applications can run in the RTR Version 3 environment. Table 6–2 provides information on application interoperability.

Table 6–2 Application Interoperability

Application Feature	Description
Mixing Version 2 and Version 3 style calls	You cannot mix RTR Version 2 and RTR Version 3 calls in the same application.
Transaction identification	In RTR Version 3, unless your application uses only the RTR Version 2 API and DECnet, the size of the transaction identification is larger than in RTR Version 2 (28 bytes compared to 8 bytes).
Version 2 and Version 3 applications talking to one another	RTR Version 2 and RTR Version 3 applications can directly communicate using RTR calls.
The DSDEF feature for data marshalling	A new feature in RTR Version 3 is DSDEF, with which an application can specify data marshalling requirements. With RTR Version 3, you can specify data format and RTR Version 3 can do format translations where required. See the Reliable Transaction Router *Application Programmer's Reference Manual* for more information.

6.5 Support for $GETTXI

The $GETTXI system service to return transaction information is not available in RTR Version 3. The RTR Version 3 equivalent is rtr_request_info. Applications that used $GETTXI must either remove $GETTXI or convert to RTR Version 3 calls. This change was required because of significant change to RTR data structures between RTR Version 2 and RTR Version 3.

6.6 Threaded Applications

With RTR Version 3, applications that rely on threading may not work exactly the same way they worked with RTR Version 2 on OpenVMS.

Applications that use kernel threads with RTR Version 2 will not work with RTR Version 3. RTR Version 2 was thread-reentrant, but the RTR Version 2 layer in RTR Version 3 on OpenVMS and Windoes should not be called from more than one thread. When writing threaded applications, a developer should consult OpenVMS documentation for information on what can be done at AST level in a threaded application. For example, see the *DEC C Run-Time Library Reference Manual for OpenVMS Systems*, Section 1.7.1, “Reentrancy”, for restrictions on the simultaneous use of OpenVMS ASTs and threads.

The RTR Version 3 API can be called from multiple threads when linked with the multi-threaded version of the RTR shared library provided with RTR for DIGITAL UNIX, Windows NT/95, Sun Solaris, or AIX.

6.7 DDTM Support

RTR Version 2 provides limited support for DDTM (DECdtm™) in RTR Version 3.1D and later.

6.8 Current Issues

Certain server applications that worked with RTR Version 2 may not work correctly with RTR Version 3. In RTR Version 3, server applications must have outstanding $DEQ_TX calls for each server channel at all times. RTR Version 2 could tolerate breaking of this condition; RTR Version 3 does not.

7 Performance Tips

With RTR Version 3, there are several considerations for improving performance. These are described in the following sections.

7.1 Process Quotas

OpenVMS process quotas should be increased to accomodate the use of mailboxes for processes. Check the RTR *Installation Guide* for the specific formula to use.

7.2 Journal Sizing

With RTR, the larger the journal, the more time it takes to read it. This can affect failover in some circumstances. Due to the new journal format, journal files need to be created larger than in RTR Version 2 to accomodate the larger transaction identification implemented for RTR Version 3. Monitor the journal file periodically to make it the most effective size for your RTR environment.

7.3 RTR Startup Qualifiers

RTR startup qualifiers have changed. You no longer provide specific configuration information, but use OpenVMS quotas; RTR Version 3 manages configuration requirements.

7.4 Monitoring

Monitoring is an important tool for evaluating performance on RTR facilities and between RTR nodes. It provides you with direct feedback on the performance and characteristics of your RTR system and its applications.

Additionally, because RTR Version 3 monitoring uses RTRACP heavily, doing constant monitoring can affect performance. If monitoring is affecting performance on your RTR system, use a longer monitoring interval than the default (2 seconds). Use the RTR MONITOR command with the /INTERVAL=*seconds* qualifier to increase the monitoring interval.

You generally use a different monitoring interval when debugging a new environment, facility, or application than when you are in production. This is not different between RTR Version 2 and RTR Version 3.

7.5 Memory

RTR Version 3 requires more memory than RTR Version 2. Monitor the RTR and application processes for increased page fault rates, and increase process memory quotas if required.

7.6 Simultaneous Multiprocessing

With RTR Version 2, threaded applications could use Symmetric Multiprocessing (SMP) effectively. In RTR Version 3, SMP will not provide the performance advantages of RTR Version 2. The single control process of RTRACP in RTR Version 3 does not take advantage of an SMP configuration. Similarly, because with RTR Version 3 there is only a single command server, you cannot gain advantages associated with the ability in RTR Version 2 to run command servers on different processors. However, applications can run multiple concurrent servers as in RTR Version 2.

8 Problem Diagnosis and Reporting

RTR Version 3 provides a new error log and logical names to assist tracing errors including:

- the RTR operator log file, capturing events that occur, and useful for diagnosis of problems
- the RTR_ERROR.LOG file
- the dump file (.DMP), a binary crash dump that, if needed, must be sent to your support service

8.1 RTR Operator Log

Always initiate an operator log in your RTR$STARTUP.COM procedure. Place it in a disk partition separate from the RTR journal or database.

8.2 RTR_ERROR.LOG File

The RTR error log is new with RTR Version 3. The RTR_ERROR.LOG file captures the call stack and counter information in a form readable by a human when a crash occurs, and can be read even when no dump is available. With RTR Version 2, the only log captured was the operator log, named by the operator. The operator log remains in RTR Version 3, and the new crash dump log provides additional information.

8.3 Dump File

The system logical name RTR$DUMP_DIRECTORY points to the crash-dump directory. The logical name specified by the user can be set in RTR$STARTUP.COM.

To produce a dump, the procedure is the same as in RTR Version 2 (in unsupported mode, type DEBUG ACP to get a crash dump).

8.4 Producing and Directing a Trace

To start and stop a trace, use the SET TRACE command. To perform a trace, use the following procedure:

1. `set log/file=`*filename*	Starts your log of the trace. This captures the trace of the specified subsystem in the specified file.
2. `set mode/unsupported`	Sets mode to unsupported.
3. `set trace/subsystem=`*name*	Starts the trace. Valid names are RTR subsystem names such as API and JNL.

or `set trace/subsystem=(API, CIF, CRM)`	Starts the trace on subsystems API, CIF and CRM.
4. `set mode/nounsupported`	Sets mode to supported.
.	Trace continues.
.	Trace continues.
5. `set mode/unsupp`	Sets mode to unsupported.
6. `set trace`	Stops the trace.
7. `set mode/nounsupp`	Sets mode back to supported.

Running a trace can affect performance, so be sure to turn it off again when done (see step 6).

8.5 Dealing with a Looping Process

If your system appears to be hung, this may be caused by an RTR process looping. To analyze the problem, do the following:

1. Enter the `SHOW PROCESS RTRACP/CONTINUOUS` OpenVMS command.
2. Examine the running process, which will be in computable mode. PC samples typically assume values with a narrow range, or recur frequently.
3. Lower the priority of the ACP process.
4. Enter the OpenVMS `ANALYZE/SYSTEM SET PROCESS RTRACP` command.
5. Enter the `SHOW CALL/NEXT` command until you see PC values corresponding to RTR software.
6. If you cannot resolve the performance issues yourself, send the logs and your process output to DIGITAL Multivendor Customer Services using normal problem reporting channels.

8.6 Contents of the RTR Journal File

With RTR Version 2, the only way to examine the content of journal files was to stop the ACP. With RTR Version 3, you can do this without stopping RTR using the `DUMP JOURNAL` command.

Index